

# UTILITY PATENT APPLICATION TRANSMITTAL

(New Nonprovisional Applications Under 37 CFR § 1.53(b))

Attorney Docket No.

50325-0512

## TO THE COMMISSIONER FOR PATENTS:

Transmitted herewith is the patent application of ( ) application identifier or (X) first named inventor, David A. McGrew, et al., entitled STREAM CIPHER ENCRYPTION METHOD AND APPARATUS THAT CAN EFFICIENTLY SEEK TO ARBITRARY LOCATIONS IN A KEYSTREAM, for a(n):

(X) Original Patent Application.

( ) Continuing Application (prior application not abandoned):

( ) Continuation ( ) Divisional ( ) Continuation-in-part (CIP)  
of prior application No: \_\_\_\_\_ Filed on: \_\_\_\_\_

( ) A statement claiming priority under 35 USC § 120 has been added to the specification.

Enclosed are:

(X) Specification 32 Total Pages; (X) Formal Drawing(s); 10 Total Sheets; (X) Cover Sheet 1 Page

(X) Oath or Declaration: 2 Pages

(X) A Newly Executed Combined Declaration and Power of Attorney:

(X) Signed. ( ) Unsigned. ( ) Partially Signed.

( ) A Copy from a Prior Application for Continuation/Divisional (37 CFR § 1.63(d)).

( ) Incorporation by Reference. The entire disclosure of the prior application, from which a copy of the oath or declaration is supplied, is considered as being part of the disclosure of the accompanying application and is hereby incorporated herein by reference.

( ) Signed Statement Deleting Inventor(s) Named in the Prior Application. (37 CFR § 163(d)(2)).

( ) Power of Attorney.

(X) Return Receipt Postcard.

( ) Associate Power of Attorney.

( ) A Check in the amount of \$982.00 for the Filing Fee.

( ) Preliminary Amendment.

( ) Information Disclosure Statement and Form PTO-1449.

( ) A Duplicate Copy of this Form for Processing Fee Against Deposit Account.

( ) A Certified Copy of Priority Documents (if foreign priority is claimed).

( ) Statement(s) of Status as a Small Entity.

( ) Statement(s) of Status as a Small Entity Filed in Prior Application, Status Still Proper and Desired.

(X) Other: Assignment and Recordation Form

CLAIMS AS FILED				
FOR	NO. FILED	NO. EXTRA	RATE	FEE
Total Claims	21	1	\$18.00	\$18.00
Independent Claims	6	3	\$78.00	\$234.00
Multiple Dependent Claims (if applicable)				\$0.00
Assignment Recording Fee				\$40.00
Basic Filing Fee				\$690.00
Total Filing Fee				\$ 982.00

Charge \$ 982.00 to Deposit Account 50-1302 pursuant to 37 CFR § 1.25. At any time during the pendency of this application, please charge any fees required or credit any overpayment to this Deposit Account.

Respectfully submitted,

By:

Christopher J. Palermo,  
Attorney of Record, Reg. No. 42,056

Date: September 29, 2000

Correspondence Address:

Hickman Palermo Truong & Becker, LLP  
1600 Willow Street  
San Jose, California 95125-5106  
Telephone: (408) 414-1080  
Facsimile: (408) 414-1076

I hereby certify that this is being deposited with the U.S. Postal Service "Express Mail Post Office to Addressee" service under 37 CFR § 1.10 on the date indicated below and is addressed to:

Commissioner for Patents  
Box Patent Application  
Washington, D.C. 20231

By:

Typed Name: Casey Moore

Express Mail Label No.: EL624356549US

Date of Deposit: September 29, 2000

50325-0512  
(Seq. No. 3245 / CPOL 86661)

*Patent*

UNITED STATES PATENT APPLICATION

FOR

STREAM CIPHER ENCRYPTION METHOD AND APPARATUS THAT CAN EFFICIENTLY SEEK  
TO ARBITRARY LOCATIONS IN A KEYSTREAM

INVENTORS:

DAVID A. MCGREW  
SCOTT R. FLUHRER

PREPARED BY:

HICKMAN, PALERMO, TRUONG & BECKER  
1600 WILLOW STREET  
SAN JOSE, CA 95125  
(408) 414-1080

EXPRESS MAIL CERTIFICATE OF MAILING

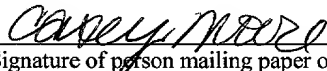
"Express Mail" mailing label number: EL624356549US

Date of Deposit September 29, 2000

I hereby certify that this paper or fee is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service under 37 CFR 1.10 on the date indicated above and is addressed to the Assistant Commissioner for Patents, Washington, D.C. 20231.

CASEY MOORE

(Typed or printed name of person mailing paper or fee)



(Signature of person mailing paper or fee)

# STREAM CIPHER ENCRYPTION METHOD AND APPARATUS THAT CAN EFFICIENTLY SEEK TO ARBITRARY LOCATIONS IN A KEYSTREAM

## FIELD OF INVENTION

The present invention generally relates to cryptography. The invention relates more  
5 specifically to a stream cipher encryption method and apparatus that can efficiently seek to  
arbitrary locations in a keystream, including a method of generating an arbitrary segment of  
keystream for use by a stream cipher in encryption or decryption.

## BACKGROUND OF THE INVENTION

Stream ciphers convert a plaintext to a ciphertext one bit at a time. In general, a  
10 stream cipher has a keystream generator that outputs a keystream consisting of a series of bits  
that, for perfect security, vary in value in an unpredictable manner. Each keystream bit is  
combined using a Boolean exclusive-OR operation (XOR) with an incoming bit of the  
plaintext, resulting in an output bit of the ciphertext. Thus, an additive stream cipher encrypts  
a plaintext by bitwise adding a pseudo-randomly generated keystream into the plaintext,  
15 modulo two.

For decryption, the ciphertext bits are XORed with an identical keystream to recover  
the plaintext bits. Accordingly, a stream cipher is ideally suited to encrypting a continuing  
stream of data, such as the data passing over a network connection between two computers or  
other network elements. Also, the security of a stream cipher resides in the randomness of the  
20 keystream, however, the keystream must be reproducible in identical form at decryption  
time. Therefore, design of the keystream generator is essential to security and practical  
operation.

FIG. 1A is a simplified block diagram of a stream cipher. A key  $K$  401 is fed to keystream generator 402, which outputs keystream 410. Plaintext 412 is encrypted by an encryption function 416 based on keystream 410. As a result, ciphertext output 414 is produced.

5        The keystream generator of such a stream cipher can be described in terms of a state update function and an output function. For example, in FIG. 1A, keystream generator 402 has internal state information 404, a next state function 406 (state update function), and output function 408. The state update function maps the internal state of the keystream generator at one instant to its next value. The output function maps the internal state to a  
10       segment of keystream, and the keystream is defined as the concatenation of the values of the output function. Further background information on stream ciphers is provided in B. Schneier, "Applied Cryptography: Protocols, Algorithms and Source Code in C," 2<sup>nd</sup> ed. (New York: John Wiley & Sons, 1996).

Block ciphers such as the Data Encryption Standard (DES) are popularly used for  
15       encryption of computer communications. However, empirical evidence indicates that stream ciphers are faster than block ciphers at equivalent security levels. For example, in practical evaluation, the stream ciphers RC4 and SEAL have been determined to be significantly faster than any secure block cipher when implemented on general-purpose computer processors. Further, RC4 and SEAL have survived years of scrutiny by cryptanalysts. SEAL is  
20       described in U.S. Pat. No. 5,454,039; U.S. Pat. No. 5,675,652; U.S. Pat. No. 5,835,597; Rogaway, P. and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Proceedings of the 1994 Fast Software Encryption Workshop, Lecture Notes In Computer Science, Volume 809, Springer-Verlag, 1994, pp. 56-63; Rogaway, P. and Coppersmith, D., "A Software-Optimized Encryption Algorithm", Journal of Cryptology, Volume 11, Number  
25       4, Springer-Verlag, 1998, Pages 273-287, and at

<http://www.cs.ucdavis.edu/~rogaway/papers/seal-abstract.html>. Both SEAL and RC4 are discussed in Schneier.

Further, theoretically, a stream cipher is inherently immune to a chosen plaintext attack, and can contain more state information than a block cipher. A block cipher needs to  
5 have both encryption and decryption to be secure, and needs to have the avalanche property from the middle to both ends. For example, changing a single bit in the middle of the cipher should change each bit of the input and the output with probability of about 1/2. Also, the stream cipher has the advantage that its outputs are ordered, while a block cipher must be able to efficiently compute every possible output in any possible order. As a result, for many  
10 applications stream ciphers are now clearly preferable over block ciphers.

Unfortunately, many stream ciphers have a significant limitation; most cannot efficiently seek to an arbitrary location in their keystream. In this context, seeking to an arbitrary location in the keystream means generating a segment of keystream that is conceptually located an arbitrary number of bits ahead of that portion of keystream that  
15 would be generated by ordinary operation of the keystream in its then-current state. This capability is required for numerous practical applications. For example, in a communications protocol that uses unreliable transport, there is no guarantee that data packets of a particular flow will arrive in order, or arrive at all. Examples of such protocols include Internet Protocol (IP), UDP, and RTP. Such protocols commonly experience loss and reorder of  
20 packets in practice. Therefore, for a flow that includes successive packets *a*, *b*, and *c*, a cipher may need to encrypt packet *c* before it encrypts packet *b*. A stream cipher can be used to provide privacy for data communicated using such protocols, if the cipher can seek to the proper location in the keystream for packet *c* based on a sequence number.

Similarly, an encrypted disk partition or file system can use a stream cipher if the  
25 cipher supports the seek operation.

These examples do not require the random access capability of a block cipher, in which all inputs are equally simple to compute. Rather, the example applications require the capability to seek into the keystream, with a seek time that is not significant relative to the time required to generate the keystream itself. In this context, "seek" is used in the same sense as used in the POSIX and ANSI C functions for repositioning the offset of a file descriptor.

In one past approach to providing a stream cipher with a seek capability, the state update function is made linear in some field. In this approach, a seek is a composition of linear operations, and therefore is itself linear. This approach is similar to using a block cipher in counter mode, which imposes requirements on the output function that are similar to the requirements on block ciphers.

In an alternative approach, as taken by Rogaway et al. in the design of the SEAL cipher, a special seek function is defined that pseudo-randomly maps an index and a fixed key to an internal state of a keystream generator. Based on this state information, the keystream generator can generate a length of keystream. The keystream for the cipher is defined to be the concatenation of the keystreams generated for each index, with indices in ascending order. Effectively, this approach creates a stream cipher that can seek to some regularly spaced locations in its keystream.

While this approach is satisfactory for many applications, some applications may require the ability to seek to an arbitrary location in the keystream. For example, an encrypted database containing many small records could have this requirement. In addition, the seek function approach adds security requirements. The seek function itself must be secure, and the seek and advance functions must be such that they do not interact in an insecure way.

Based on the foregoing, there is a clear need for an additive stream cipher method that can seek to an arbitrary location in its keystream.

There is a specific need for a stream cipher that provides a keystream seek capability without using a linear state update function, and without a special seek function.

There is also a need to provide such a stream cipher in an embodiment that achieves excellent performance when executed in software implemented for general-purpose computer  
5 processors.

## SUMMARY OF THE INVENTION

The foregoing needs, and other needs and objects that will become apparent for the following description, are achieved in the present invention, which comprises, in one aspect, a method of automatically generating a keystream segment of an arbitrary location of a complete keystream of an additive stream cipher. A location value that identifies a location of the keystream segment within the complete keystream is received. A state value for a leaf node of a balanced binary tree is created and stored, wherein the leaves of the tree represent the complete keystream and the leaf node represents the keystream segment at the location, by a traversal of the tree from root node to the leaf node wherein a leftward tree branch transition comprises computing a first non-linear function and a rightward tree branch transition comprises computing a second non-linear function. The keystream segment is created and stored by applying a third function to the state value of the leaf node. The entire keystream can be efficiently computed, in order, via a preorder traversal of the tree.

According to another aspect, the invention provides a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream. A segment of plaintext is received, along with information indicating where the segment of plaintext is located within a complete plaintext. Based on the location information, a keystream segment for that location is created and stored. The segment of plaintext is then enciphered by combining the keystream segment with the plaintext segment using an exclusive OR operation, resulting in creating and storing a segment of ciphertext.

In other aspects, the invention encompasses a computer apparatus, a computer readable medium, and a carrier wave configured to carry out the foregoing steps.



## BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

5        FIG. 1A is a simplified block diagram of a stream cipher.

FIG. 1B is a block diagram of a portion of a balanced binary tree that may be used in one embodiment.

FIG. 1C is a flow diagram of a method of generating a segment of a keystream.

FIG. 1D is a flow diagram of a method of encrypting network communications.

10       FIG. 2 is a computational graph diagram that illustrates one embodiment of a function  $c \circ f$ .

FIG. 3 is a computational graph diagram that illustrates one embodiment of a function  $c \circ g$ .

15       FIG. 4 is a computational graph diagram that illustrates a concatenation of FIG. 2 and FIG. 3.

FIG. 7 is a block diagram that illustrates a computer system upon which an embodiment may be implemented.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream, are described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

### COMPUTATION TREE APPROACH

An improved additive stream cipher is defined in part by a keystream generating approach that uses a balanced binary tree.

FIG. 1B is a block diagram of a portion of a balanced binary tree that may be used in one embodiment. In the example tree 100 of FIG. 1, root node 102 has fixed state information. Each intermediate node 104 provides a transition path from root node 102 to two of a plurality of leaf nodes 106. Each leaf node 106 represents and is associated with a small length of keystream.

Computation of a keystream segment at a specified location within the complete keystream is carried out by computing from the root node 102 to one of the leaf nodes 106 that is associated with the specified location, then computing the rest of the nodes using a pre-order tree traversal. Thus seeking to a location in the keystream is equivalent to computing the state for a particular leaf 106 in the tree 102, a computation that requires computational effort that is logarithmic in the size of the keystream.

The root node 104 and each leaf node 106 contain  $m$  bits of state information. A left branch of tree 102 is computed using the nonlinear function

$$a: GF(2^m) \rightarrow GF(2^m)$$

and a right branch is computed using the nonlinear function

5 
$$b: GF(2^m) \rightarrow GF(2^m)$$

The outputs of the cipher are computed by applying the function

$$c: GF(2^m) \rightarrow GF(2^r)$$

to the state information of the leaf nodes, which maps the  $n$  internal bits to an output of size  $r$ . Accordingly, in FIG. 1B, leftward transitions from node to node have reference label  $a$  to indicate that such transitions involve computing function  $a$ . Similarly, rightward transitions from node to node have reference label  $b$  to indicate that such transitions involve computing function  $b$ . Exit paths from leaf nodes 106 have reference label  $c$  to indicate that such paths involve computing function  $c$ .

Generating a complete keystream in order is done by a preorder traversal of the entire tree, starting at the root. A binary tree with height  $h$  contains  $2^h - 1$  nodes,  $2^{h-1}$  leaves, and  $2^h - 1 - 2$  edges. In this context, a tree with one node is defined as having height one. Each edge is computed exactly once in a traversal of the tree. Therefore, the present approach can generate keystream at a rate of approximately:

$$\frac{r2^{h-1}}{R_c2^{h-1} + R_{ab}(2^h - 2)} \cong \frac{r}{R_c + 2R_{ab}}$$

20 where  $R_c$  is the rate at which the function  $c$  can be evaluated, and  $R_{ab}$  is the average rate at which the functions  $a$  and  $b$  can be evaluated.

In one embodiment, the tree is implemented by creating and storing a stack in memory of  $h$  elements, the  $i^{th}$  element of which contains the state information of the node on the  $i^{th}$  path from the root to the leaf corresponding to the current output. Thus, the computation tree defines a cipher with  $mh$  bits of internal state, wherein the state update

25

function modifies only a portion of the current state. An important implementation consideration is that the preorder traversal is readily adaptable to parallel processing.

The functions *a* and *b* are comparable to the round functions of a substitution-permutation network. They are the main source of non-linearity in the cipher disclosed herein. The function *c* can be used to hide the internal state, as it has fewer outputs than inputs.

#### CIPHER PROCESSING FUNCTIONS AND KEY SETUP APPROACH

In one embodiment, a stream cipher that can seek to an arbitrary location in its  
10 keystream involved computing functions *a*, *b*, and *c*, and a key setup approach.

#### -- IMPROVED CIPHER PROCESSING APPROACH

Some general-purpose central processing units, such as the Intel x86 family of processors, have no more than seven (7) available registers. In one specific embodiment, a cipher approach is optimized to operate with seven registers. Accordingly, in this  
15 embodiment, the present approach achieves high throughput on general-purpose processors.

FIG. 1C is a flow diagram of a method of generating a segment of a keystream. In block 120, a location value that identifies a location of a keystream segment within a complete keystream is received. In data communication applications, the location value may comprise a packet identifier that indicates the relative order, within a flow of packets, of a  
20 packet that has been received out of order with respect to other packets in the flow.

In block 122, a state value is created and stored by conceptually traversing a balanced binary tree that represents the complete keystream. As stated in block 124, a leaf node in the tree represents the keystream segment. Traversal proceeds from the root node to the leaf node associated with the specified keystream segment, as shown in block 126. To traverse a left  
25 branch, a first non-linear function is computed, as shown by block 128. To traverse a right branch, a second non-linear function is computed, as shown by block 130.

In block 132, a third function is applied to the state value, to result in creating and storing the keystream segment.

FIG. 1D is a flow diagram of a method of encrypting network communications.

5 In block 140, an unencrypted packet is received as part of a stream of network communications. Block 140 may involve receiving one or more packets at any suitable network device or end station, such as a router, switch, gateway, server, etc. Also, the unencrypted information may comprise a portion of a packet, e.g., a payload of a packet wherein the header is treated separately.

10 In block 142, a location of the packet within a network flow is determined. Block 142 may involve, for example, determining the order of a particular packet within a larger flow of multiple packets, where the packets are received out of order.

In block 144, a keystream segment corresponding to the packet location is created and stored. Block 144 may involve carrying out the steps of FIG. 1C in order to generate a segment of keystream appropriate for the received packet.

15 In block 146, the packet is encrypted or enciphered using the keystream segment as applied to a stream cipher function. Block 146 may involve encrypting only the payload of a packet.

The process of FIG. 1D and the other processes described herein are applicable to encryption of packets that form portions of voice over Internet Protocol flows in network communications as part of the IPsec protocol, and for the encryption of unreliable network traffic in general, for example, traffic transported using the UDP protocol.

25 In one embodiment, the number  $m$  of bits of state per node, and the key size, are specified by parameter values. The parameterizable width enables reduced size versions of the cipher to be studied, enabling numerical experiments on the cipher to be more meaningful. The parameterizable key size is useful to adapt the cipher to different security policies, and to protect against typical plaintext attacks by enabling keys to be increased in

size to compensate for the possibility of cryptographic attacks that use precomputation. See, e.g., D. McGrew et al., "Attacks on Linearly Redundant Plaintext and Implications on Internet Security," the Proceedings of the Seventh Annual Workshop on Selected Areas in Cryptography.

- 5           Accordingly, in one embodiment,  $m$  is restricted to being a multiple of twelve, because of the internal structure of the functions  $a$  and  $b$ . In the description herein,  $x$ ,  $y$  and  $z$  denote  $n$  bit quantities, and each node 104, 106 of tree 100 has  $m=3n$  bits of state denoted as  $z|y|x$ , where the symbol  $|$  denotes concatenation, and  $n$  is a multiple of four

- In one embodiment, the state contained by each node 102, 104, 106 of tree 100  
10   contains a unique node number to ensure that two distinct nodes cannot have the same state. The node number is denoted as  $z$ , and counted as part of the state of the node to which it corresponds. The node number of the root node 102 is defined as one. If the node number of a node is  $z$ , that of its left child is  $2z$  and that of its right child is  $2z+1$ . Functions of the cipher use substitution, rotation, and addition modulo  $2^n$ . The "filter" function  $c$  is defined as the  
15   linear reduction of  $2n$  bits to 2 bits. The branching functions  $a$  and  $b$  are defined as the composition of a diffusion function  $d$  with the nonlinear "confusion" functions  $f$  and  $g$ . Thus,

$$a = f \circ d$$

and

$$b = g \circ d$$

- 20   The definitions of functions  $f$ ,  $g$ ,  $d$ , and  $c$  are

$$f(z|y|x) = 2z | S(R(S(R(y)))) | L(S(L(S(x))))$$

$$g(z|y|x) = 2z+1 | L(S(L(S(y)))) | S(R(S(R(\Box x))))$$

$$d(z|y|x) = z | x + y + z | 2x + y + z$$

$$c(z|y|x) = x \oplus y$$

- 25   where integer addition modulo two to the power  $n$  is denoted as  $+$ , bitwise exclusive-or is denoted as  $\oplus$ , and bitwise complementation is denoted as  $\Box$ . The functions  $R$  and  $L$  indicate

rotation by  $n/4$  bits and to the right and left, respectively (where “right” means in the direction of the least significant bit). The nonlinear function  $S$  is implemented as a key-dependent substitution table.

The filter function  $c$  is a bitwise exclusive or of  $x$  and  $y$ . The composition of  $c$  with  $f$  and  $g$  is shown in FIG. 2, FIG. 3. Specifically, FIG. 2 is a computational graph diagram that illustrates one embodiment of the function  $c \circ f$ , as indicated by reference numeral 200, and FIG. 3 is a computational graph diagram that illustrates one embodiment of the function  $c \circ g$ , designated by reference numeral 300. FIG. 4 is a computational graph diagram that illustrates a concatenation of FIG. 2 and FIG. 3 (reference numeral 400). FIG. 5 is a computational graph diagram that illustrates one embodiment of the function  $f \circ d$  (reference numeral 500). FIG. 6 is a computational graph diagram that illustrates one embodiment of the function  $g \circ d$ , as indicated by reference numeral 600.

In FIG. 2, FIG. 3, FIG. 4, FIG. 5, and FIG. 6, inputs  $X0, X1, X2, X3$  represent the bytes of  $x$ , where  $X0$  is the least significant byte,  $X1$  is the second least significant, and so on. Similarly, inputs  $Y0, Y1, Y2, Y3$  represent the bytes of variable  $y$ , from least significant to next least significant, and inputs  $Z0, Z1, Z2, Z3$  represent bytes of variable  $z$ . The function  $S$  depends only on the lowest  $n/4$  bits of its argument, and can be viewed as four component functions  $S0, S1, S2, S3$ , as indicated by like labeled blocks in FIG. 2, FIG. 3, FIG. 4, FIG. 5, and FIG. 6. A key setup approach is defined such that component functions  $S0, S1, S2, S3$  are invertible.

FIG. 2 and FIG. 3 illustrate how the output of the cipher disclosed herein is derived from the state of the leaf nodes using one embodiment of  $f, g$ , and  $c$ . The structures of  $f, g$ , and  $c$  are optimized, in this embodiment, to resist up and down attacks. Referring to FIG. 3, the variables at the top and bottom ( $z, w$ ) represent two adjacent  $n$ -bit outputs of the cipher. The state at the middle of the network, represented by  $x, y$ , is the output of the function  $d$  for each of the leaves.

The manner in which the functions  $f$  and  $g$  use a substitution table is similar to that of the block cipher WAKE, which is described in B. Schneier, *supra*, at 400,. However, in the approach described herein, greater parallelism and lower register usage is achieved by using the substitution table to modify the value of its argument.

5           -- KEY SETUP

Key setup consists of generating four invertible functions, and storing them in concatenated form in a substitution table. The substitution operation herein maps a single variable  $x$  to  $x \oplus s(x)$ , for some substitution table  $s$  that depends only on the lowest 8 bits of  $x$ . Thus, invertibility requires that the function mapping the lowest 8 bits of  $x$  to lowest 8 bits  
10 of  $x \oplus s(x)$  be invertible. This is achieved by pseudorandomly selecting an invertible function mapping 8 bits to 8 bits, then composing that function with the bitwise exclusive-or function. Invertible functions are pseudorandomly generated by swapping elements at random in the table, an approach similar to that of the RC4 next-state function.

Detailed pseudocode for this method follows, in which  $S0$ ,  $S1$ ,  $S2$ , and  $S3$  are the  
15 substitution tables described above, 'key' represents the raw key, and  $i$ ,  $j$ ,  $k$ , and 'index' are integers between zero and 255. The symbol % denotes the modulus operator. The number of bytes in a key is represented here by the symbol `bytes_in_key`, which is a parameter of this method.

PSEUDOCODE FOR KEY SETUP

20           1. (initialize substitution tables to zero)  
          for  $i$  from 0 to 255  
              set  $S0[i]$  to 0  
              set  $S1[i]$  to 0  
25           set  $S2[i]$  to 0  
              set  $S3[i]$  to 0  
          endfor



2. (set substitution tables to key-dependent permutations)

for j from 0 to 255

for i from 0 to 255

set S3[i] to S2[i]

5 set S2[i] to S1[i]

set S1[i] to S0[i]

set S0[i] to i

endfor

set index to j

10 for k from 1 to num\_passes

for i from 0 to 255

set index to index + (key[i % bytes\_in\_key] + S0[i]) % 255

set tmp to S0[i]

set S0[i] to S0[index]

15 set S0[index] to tmp

set tmp to S1[i]

set S1[i] to S1[index]

set S1[index] to tmp

set tmp to S2[i]

20 set S2[i] to S2[index]

set S2[index] to tmp

set tmp to S3[i]

set S3[i] to S3[index]

set S3[index] to tmp

25 endfor

endfor

endfor

3. (set S0 equal to the xor of itself with the identity permutation)

30 for i from 0 to 255

set S0[i] to the value of S0[i] exclusive-ored with i

endfor

C language computer program source code for an example of a key setup approach is presented in Appendix A.

#### -- PERFORMANCE

- 5       The functions  $f$  and  $g$  can each be computed in seven instructions on a general purpose CPU that can issue two instructions simultaneously, using five registers (holding  $x$ ,  $y$ ,  $z$ , a temporary variable, and a pointer to the substitution table). The theoretical encryption rate is about fifteen cycles per byte (using two execution units).

10

#### HARDWARE OVERVIEW

FIG. 7 is a block diagram that illustrates a computer system 700 upon which an embodiment of the invention may be implemented.

- Computer system 700 includes a bus 702 or other communication mechanism for communicating information, and a processor 704 coupled with bus 702 for processing
- 15   information. Computer system 700 also includes a main memory 706, such as a random access memory ("RAM") or other dynamic storage device, coupled to bus 702 for storing information and instructions to be executed by processor 704. Main memory 706 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 704. Computer system 700 further includes a read
- 20   only memory ("ROM") 708 or other static storage device coupled to bus 702 for storing static information and instructions for processor 704. A storage device 710, such as a magnetic disk or optical disk, is provided and coupled to bus 702 for storing information and instructions.

- Computer system 700 may be coupled via bus 702 to a display 712, such as a cathode
- 25   ray tube ("CRT"), for displaying information to a computer user. An input device 714, including alphanumeric and other keys, is coupled to bus 702 for communicating information

and command selections to processor 704. Another type of user input device is cursor control 716, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 704 and for controlling cursor movement on display 712. This input device typically has two degrees of freedom in two  
5 axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 700 for performing a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream. According to one  
10 embodiment of the invention, performing a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream, are provided by computer system 700 in response to processor 704 executing one or more sequences of one or more instructions contained in main memory 706. Such instructions may be read into main memory 706 from another  
15 computer-readable medium, such as storage device 710. Execution of the sequences of instructions contained in main memory 706 causes processor 704 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware  
20 circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 704 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks,  
25 such as storage device 710. Volatile media includes dynamic memory, such as main memory 706. Transmission media includes coaxial cables, copper wire and fiber optics, including the

wires that comprise bus 702. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other  
5 optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 704 for execution. For example, the  
10 instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 700 can receive the data on the telephone line and use an infra-red transmitter to convert the data to an infra-red signal. An infra-red detector can receive the data carried in the infra-red signal and  
15 appropriate circuitry can place the data on bus 702. Bus 702 carries the data to main memory 706, from which processor 704 retrieves and executes the instructions. The instructions received by main memory 706 may optionally be stored on storage device 710 either before or after execution by processor 704.

Computer system 700 also includes a communication interface 718 coupled to bus  
20 702. Communication interface 718 provides a two-way data communication coupling to a network link 720 that is connected to a local network 722. For example, communication interface 718 may be an integrated services digital network ("ISDN") card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 718 may be a local area network ("LAN") card to  
25 provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 718 sends and receives

electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 720 typically provides data communication through one or more networks to other data devices. For example, network link 720 may provide a connection through local network 722 to a host computer 724 or to data equipment operated by an Internet Service Provider ("ISP") 726. ISP 726 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 728. Local network 722 and Internet 728 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 720 and through communication interface 718, which carry the digital data to and from computer system 700, are exemplary forms of carrier waves transporting the information.

Computer system 700 can send messages and receive data, including program code, through the network(s), network link 720 and communication interface 718. In the Internet example, a server 730 might transmit a requested code for an application program through Internet 728, ISP 726, local network 722 and communication interface 718. In accordance with the invention, one such downloaded application provides for carrying out a stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream.

The received code may be executed by processor 704 as it is received, and/or stored in storage device 710, or other non-volatile storage for later execution. In this manner, computer system 700 may obtain application code in the form of a carrier wave.

#### OTHER EMBODIMENTS AND MODIFICATIONS

Accordingly, a fast stream cipher that can efficiently seek to arbitrary locations in its keystream has been described. Unlike RC4, which cannot seek at all, and SEAL, which can

seek only to a limited number of locations in the keystream, the approach described herein can seek to any location in the keystream. The cipher can efficiently seek to arbitrary locations in its keystream despite the fact that it does not use a linear state update function. This property is apparent from the architecture of the cipher in which its computation graph  
5 may be represented by a tree. Keystream can be efficiently generated using a preorder traversal of the tree, and the tree structure lends itself to parallelization.

Implementations may be embodied in one or more hardware circuits, one or more field programmable gate arrays (FPGAs), or in one or more software programs or processes. Any such implementation may be parallelized, offering significant gains in performance, as a  
10 result of the tree structure of the computations.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative  
15 rather than a restrictive sense.

---

## Appendix A: Source Code

The ANSI C source code for the functions  $f$ ,  $g$ ,  $d$  and  $h$  is given below, in which each function is implemented using a macro. In these definitions,  $a$  and  $b$  are 32-bit unsigned integers (or `uint32_ts`, in POSIX terminology), and the function  $F$  is represented by the array `F[256]` of 32-bit unsigned integers. The macros `ROT8` and `ROT24` implement rotation by eight bits and twenty-four bits, respectively, where the direction of rotation is towards the most significant bit.

```
10  #define ROT8(x)  (((x) << 8) | ((x) >> 24))
    #define ROT24(x) (((x) << 24) | ((x) >> 8))

    #define f(x, y, z, F) ( \
        z += z, \
15    y = ROT24(y), \
        x ^= F[x & 0xFF], \
        y ^= F[y & 0xFF], \
        y = ROT24(y), \
        x = ROT8(x), \
20    x ^= F[x & 0xFF], \
        y ^= F[y & 0xFF], \
        x = ROT8(x) \
    )

25  #define g(x, y, z, F) ( \
        z += (z+1), \
        x = ~x; \
        x = ROT24(x), \
        x ^= F[x & 0xFF], \
30    y ^= F[y & 0xFF], \
        x = ROT24(x), \
        y = ROT8(y), \
        x ^= F[x & 0xFF], \
        y ^= F[y & 0xFF], \
35    y = ROT8(y) \
    )

    #define d(x, y, z) ( \
        x += z, \
40    y += x, \
        x += y \
    )

    #define h(a, b) (a ^ b)
```

The source code to produce the  $j$ th word of output (that is, bytes  $4j$  through  $4j+3$ ) is given below, where  $j$  is represented by the variable `leaf_num`.

```

5  uint32_t
   leviathan_output(int leaf_num) {
       int i;
       uint32_t x, y, z;

10     i = 1 << (LEVIATHAN_HEIGHT-1);
       x = y = 0;
       z = 1;

       while (i > 0 ) {
15         d(x, y, z);
           if (i & leaf) {
               g(x, y, z, F); /* right */
           } else {
               f(x, y, z, F); /* left */
20         }
           i >>= 1;
       }

       return h(x, y);
25 }

```

Source code for an embodiment of a key setup routine follows. Here, `key` is a pointer to an unsigned character string of length `bytes_in_key`, and `F` is an array of `TABLE_SIZE` words.

```

30  #define TABLE_SIZE 256
     #define NUM_PASSES 2

35  void init_leviathan_key(const unsigned char *key,
                           size_t bytes_in_key, word *F) {
       int i, j, k, index;
       word tmp;

40     for (i=0; i<TABLE_SIZE; i++)
         F[i] = 0;

       /*
45      * Each iteration of this loop we form the permutation of one line
       * (and, incidentally, also permute previously formed lines)

```



```

*/
    for (j=0; j<4; j++) {
        /*
5       * Initialize the new line to the identity permutation, and
        * shift the existing lines over one
        */
        for (i=0; i<TABLE_SIZE; i++)
            F[i] = F[i] * TABLE_SIZE + i;
10
        /*
        * Initialize index to a line-dependant value, so that the
        * four lines will get distinct permutations
        */
15       index = j;

        /*
        * Do the byte-swapping NUM_PASSES times, using the new
        * line as the index
20       */
        for (k=0; k<NUM_PASSES; k++) {
            for (i=0; i < TABLE_SIZE; i++) {
                index += (key[i % bytes_in_key] + F[i]);
                index &= (TABLE_SIZE-1);
25                 tmp = F[i];
                F[i] = F[index];
                F[index] = tmp;
            }
        }
30     }

    /*
    * Finally, set S0 equal to the xor of itself with the
    * identity permutation, so that (S0[x] ^ x) is a permutation.
35     */
    for (i=0; i < TABLE_SIZE; i++)
        F[i] ^= i;

40 }

```

## Appendix B: Test Vectors

The key represented by the hexadecimal number FA57C5C0C0DE produces the following keystream (presented as a left to right, top to bottom list of 32 bit hexadecimal numbers):

5

0x1861600e, 0x88244832, 0x2a6d8201, 0xffd0f37d,  
0xb8767ce6, 0xe7bd8954, 0xb3fc97f0, 0xe88caba1

## CLAIMS

What is claimed is:

- 1 1. A method of automatically generating a keystream segment of an arbitrary location of  
2 a complete keystream of an additive stream cipher, the method comprising the  
3 computer-implemented steps of:  
4 receiving a location value that identifies a location of the keystream segment within  
5 the complete keystream;  
6 creating and storing a state value for a leaf node of a balanced binary tree, wherein the  
7 tree represents the complete keystream and the leaf node represents the  
8 keystream segment at the location, by a preorder traversal of the tree from root  
9 node to the leaf node wherein a leftward tree branch transition comprises  
10 computing a first non-linear function and a rightward tree branch transition  
11 comprises computing a second non-linear function;  
12 creating and storing the keystream segment by applying a third function to the state  
13 value of the leaf node.
- 1 2. A method as recited in Claim 1, further comprising the steps of creating and storing  
2 the balanced binary tree by creating and storing a stack of  $h$  elements wherein the  $i^{\text{th}}$   
3 element of said stack stores a state datum for the  $i^{\text{th}}$  node on a path from a root node of  
4 the tree to the leaf node.
- 1 3. A method as recited in Claim 2, wherein the step of creating and storing a state value  
2 for a leaf node comprises the steps of computing and storing a state value for the leaf  
3 node that is unique with respect to any other state value that is computed at any other  
4 time for any other leaf node of the tree.

- 1 4. A method as recited in Claim 2, wherein the steps of computing a first non-linear  
2 function and computing a second non-linear function comprises the steps of  
3 computing first and second non-linear functions that are selected such that a set of all  
4 state values of all leaf nodes is indistinguishable from a random value.
- 1 5. The method as recited in Claim 1, wherein each leaf node stores  $n$  bits of state  
2 information, wherein  $n$  is a multiple of four.
- 1 6. The method as recited in Claim 1, further comprising the steps of:  
2 creating and storing  $3n$  bits of state information in each leaf node comprising a  
3 concatenation of three  $n/2$  bit quantities  $z|y|x$ , wherein  $n$  is a multiple of  
4 four;  
5 computing the first non-linear function  $a$  and the second non-linear function  $b$  as the  
6 composition of a diffusion function  $d$  with the nonlinear "confusion" functions  
7  $f$  and  $g$ , wherein  $a = f \circ d$  and  $b = g \circ d$  and wherein  
8  $f(z|y|x) = 2z|S(R(S(R(y))))|L(S(L(S(x))))$   
9  $g(z|y|x) = 2z+1|L(S(L(S(y))))|S(R(S(R(\neg x))))$   
10  $d(z|y|x) = z|x+y+z|2x+y+z$   
11  $c(z|y|x) = x \oplus y$   
12 wherein integer addition modulo two is denoted as  $+$ , bitwise exclusive-or is denoted  
13 as  $\oplus$ , and bitwise complementation is denoted as  $\neg$ ;  
14 wherein the  $R$  denotes rotation by  $n/4$  bits to in a direction of a least significant bit and  
15  $L$  denotes rotation by  $n/4$  bits in a direction of a most significant bit; and  
16 wherein a nonlinear function  $S$  comprises a lookup in a key-dependent substitution  
17 table.
- 1 7. The method as recited in Claim 1, wherein the third function comprises computing a  
2 linear reduction of  $n$  bits of the state value to  $n/2$  bits thereof.

- 1 8. A method as recited in Claim 6, wherein the third function comprises computing a  
2 bitwise Boolean exclusive OR of  $x$  and  $y$ .
- 1 9. A method as recited in Claim 6, further comprising the steps of creating and storing  
2 the substitution table  $S$  by selecting four invertible functions and storing the four  
3 invertible functions in a concatenated form.
- 1 10. A method as recited in Claim 6, further comprising the steps of computing functions  $f$   
2 and  $g$  in seven instructions of a central processing unit that can issue two instructions  
3 simultaneously, by using five registers to store values of  $x$ ,  $y$ ,  $z$ , a temporary variable,  
4 and a pointer to the substitution table  $S$ .
- 1 11. A method as recited in Claim 6, wherein the substitution table  $S$  comprises an array of  
2 randomly selected integer values.
- 1 12. A method as recited in Claim 6, wherein the substitution table  $S$  comprises an array of  
2 256 randomly selected 32-bit unsigned integer values.
- 1 13. The method as recited in Claim 1, further comprising the steps of creating and storing  
2 a key for use by the first non-linear function and the second non-linear function,  
3 wherein the key comprises a table of randomly selected values.
- 1 14. The method as recited in Claim 1, further comprising the steps of creating and storing,  
2 once and at a time prior to receiving the location value, a key for use by the first non-  
3 linear function and the second non-linear function, wherein the key comprises a table  
4 of randomly selected values.

1 15. The method as recited in Claim 1, further comprising the steps of creating and storing  
2 a key in the form of a plurality of pseudo-randomly selected invertible functions,  
3 wherein each of the invertible functions maps an 8-bit portion of the state value to an  
4 8-bit quantity for use as a substitute portion of the state value.

1 16. A method as recited in Claim 1, wherein the substitution table *S* comprises a plurality  
2 of sub-tables, and wherein generating the substitution table comprises (a) setting  
3 values of the sub-tables to key-dependent permutations and (b) setting values of one  
4 of the sub-tables to an exclusive OR of itself to the identity permutation.

5 17. A method of enciphering a plaintext using at least one keystream segment at an  
6 arbitrary location of a complete keystream, the method comprising the computer-  
7 implemented steps of:  
8 receiving a segment of a plaintext;  
9 receiving a location value that identifies a location of the keystream segment within  
10 the complete keystream;  
11 creating and storing a state value for a leaf node of a balanced binary tree, wherein the  
12 tree represents the complete keystream and the leaf node represents the  
13 keystream segment at the location, by a preorder traversal of the tree from root  
14 node to the leaf node wherein a leftward tree branch transition comprises  
15 computing a first non-linear function and a rightward tree branch transition  
16 comprises computing a second non-linear function;  
17 creating and storing the keystream segment by applying a third function to the state  
18 value of the leaf node;  
19 enciphering the segment of the plaintext by combining the keystream segment with  
20 the segment of the plaintext using a Boolean exclusive OR operation to result  
21 in creating and storing a segment of ciphertext.

- 1 18. A method of encrypting an ordered plurality of packets of a network communication  
2 link using at least one keystream segment at an arbitrary location of a complete  
3 keystream, the method comprising the computer-implemented steps of:  
4 receiving a packet from among the plurality of packets;  
5 determining a location value that represents a relative location of the packet among  
6 the plurality of packets;  
7 creating and storing a state value for a leaf node of a balanced binary tree, wherein the  
8 tree represents the complete keystream and the leaf node represents a  
9 keystream segment at the relative location, by a preorder traversal of the tree  
10 from root node to the leaf node wherein a leftward tree branch transition  
11 comprises computing a first non-linear function and a rightward tree branch  
12 transition comprises computing a second non-linear function;  
13 creating and storing the keystream segment by applying a third function to the state  
14 value of the leaf node;  
15 enciphering the packet by combining the keystream segment with data of the packet  
16 using a Boolean exclusive OR operation to result in creating and storing  
17 enciphered packet data.
- 1 19. A computer-readable medium carrying one or more sequences of instructions for  
2 automatically generating a keystream segment of an arbitrary location of a complete  
3 keystream of an additive stream cipher, which instructions, when executed by one or  
4 more processors, cause the one or more processors to carry out the steps of:  
5 receiving a location value that identifies a location of the keystream segment within  
6 the complete keystream;  
7 creating and storing a state value for a leaf node of a balanced binary tree, wherein the  
8 tree represents the complete keystream and the leaf node represents the  
9 keystream segment at the location, by a preorder traversal of the tree from root  
10 node to the leaf node wherein a leftward tree branch transition comprises  
11 computing a first non-linear function and a rightward tree branch transition  
12 comprises computing a second non-linear function;

13           creating and storing the keystream segment by applying a third function to the state  
14           value of the leaf node.

1

1   20.   An apparatus for automatically generating a keystream segment of an arbitrary  
2       location of a complete keystream of an additive stream cipher, comprising:  
3       means for receiving a location value that identifies a location of the keystream  
4       segment within the complete keystream;  
5       means for creating and storing a state value for a leaf node of a balanced binary tree,  
6       wherein the tree represents the complete keystream and the leaf node  
7       represents the keystream segment at the location, by a preorder traversal of the  
8       tree from root node to the leaf node wherein a leftward tree branch transition  
9       comprises computing a first non-linear function and a rightward tree branch  
10      transition comprises computing a second non-linear function;  
11      means for creating and storing the keystream segment by applying a third function to  
12      the state value of the leaf node.

1   21.   An apparatus for automatically generating a keystream segment of an arbitrary  
2       location of a complete keystream of an additive stream cipher, comprising:  
3       a network interface that is coupled to the data network for receiving one or more  
4       packet flows therefrom;  
5       a processor;  
6       one or more stored sequences of instructions which, when executed by the processor,  
7       cause the processor to carry out the steps of:  
8       receiving a location value that identifies a location of the keystream segment  
9       within the complete keystream;

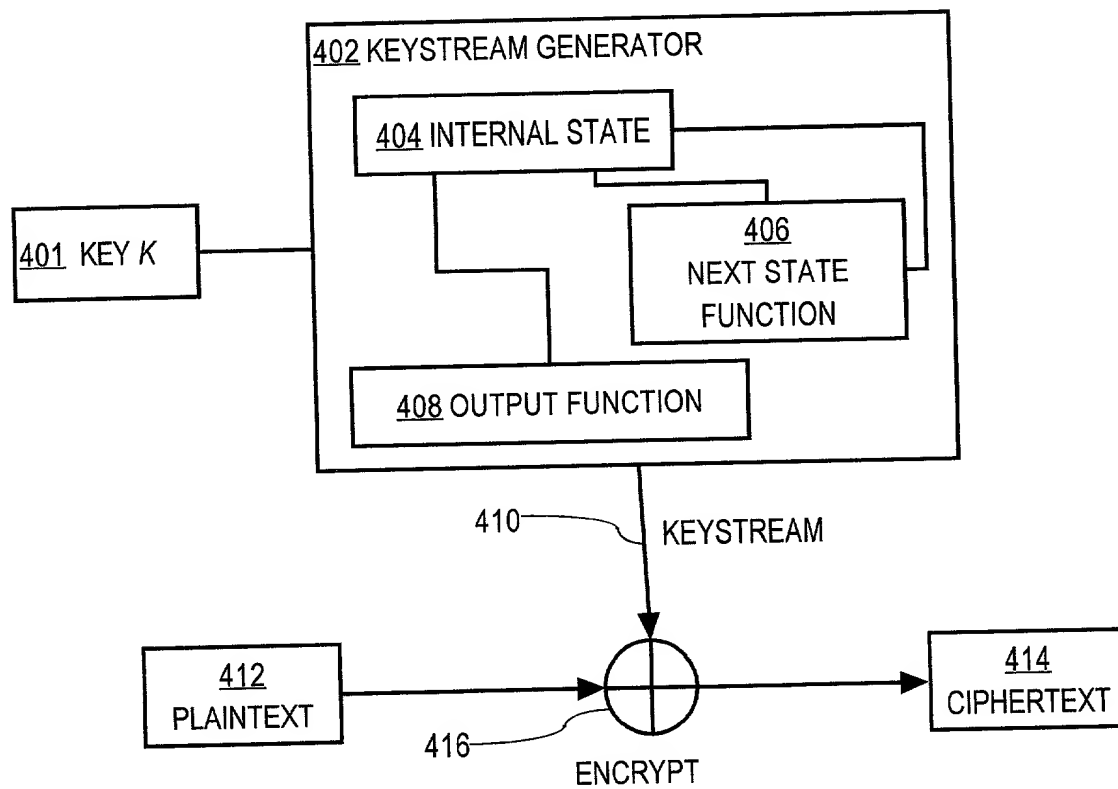


10                   creating and storing a state value for a leaf node of a balanced binary tree,  
11                   wherein the tree represents the complete keystream and the leaf node  
12                   represents the keystream segment at the location, by a preorder  
13                   traversal of the tree from root node to the leaf node wherein a leftward  
14                   tree branch transition comprises computing a first non-linear function  
15                   and a rightward tree branch transition comprises computing a second  
16                   non-linear function;  
17                   creating and storing the keystream segment by applying a third function to the  
18                   state value of the leaf node.

## ABSTRACT OF THE DISCLOSURE

A stream cipher encryption method and apparatus that can efficiently seek to arbitrary locations in a keystream, and a method of generating an arbitrary segment of keystream.

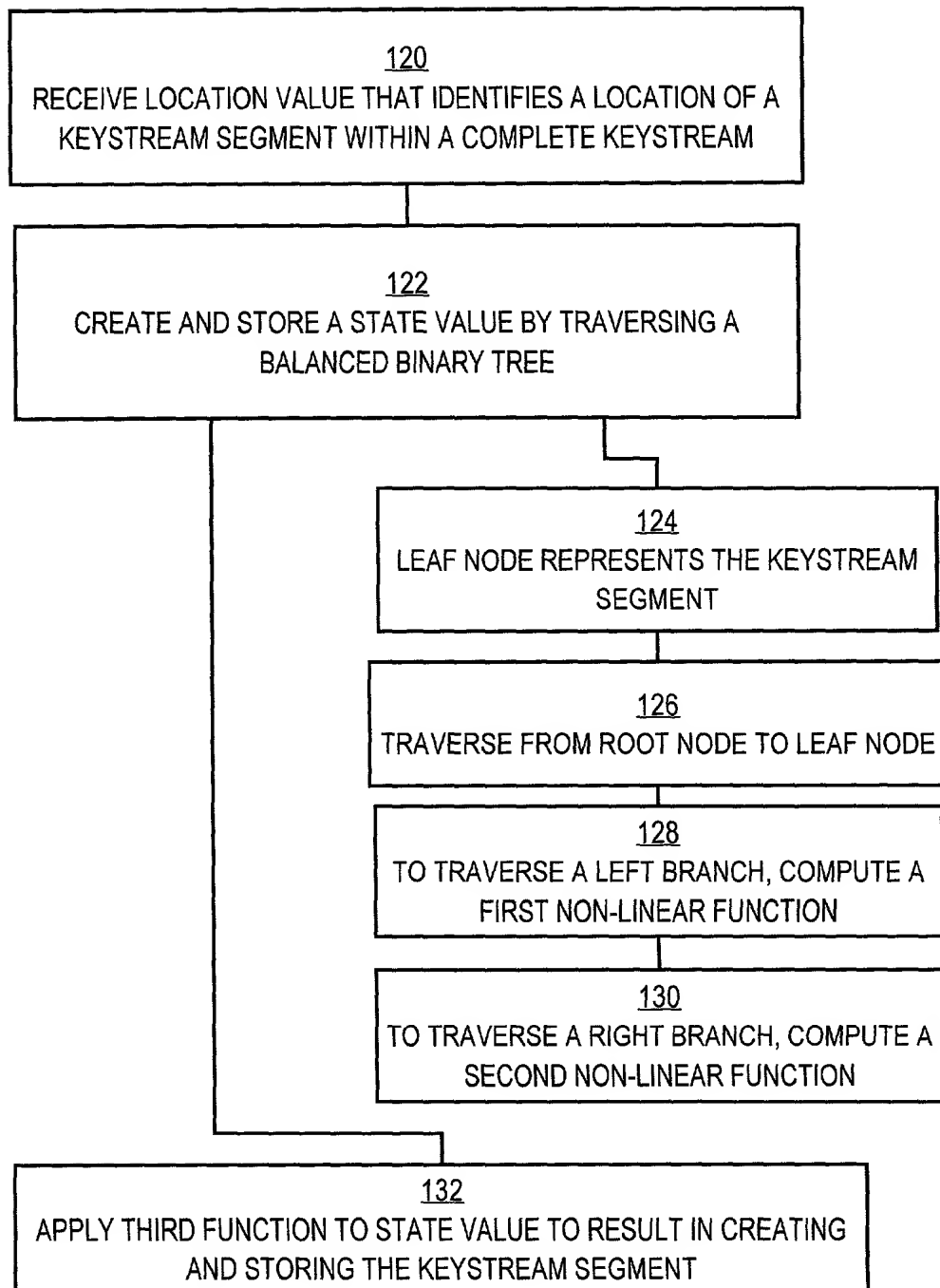
*Fig. 1A*



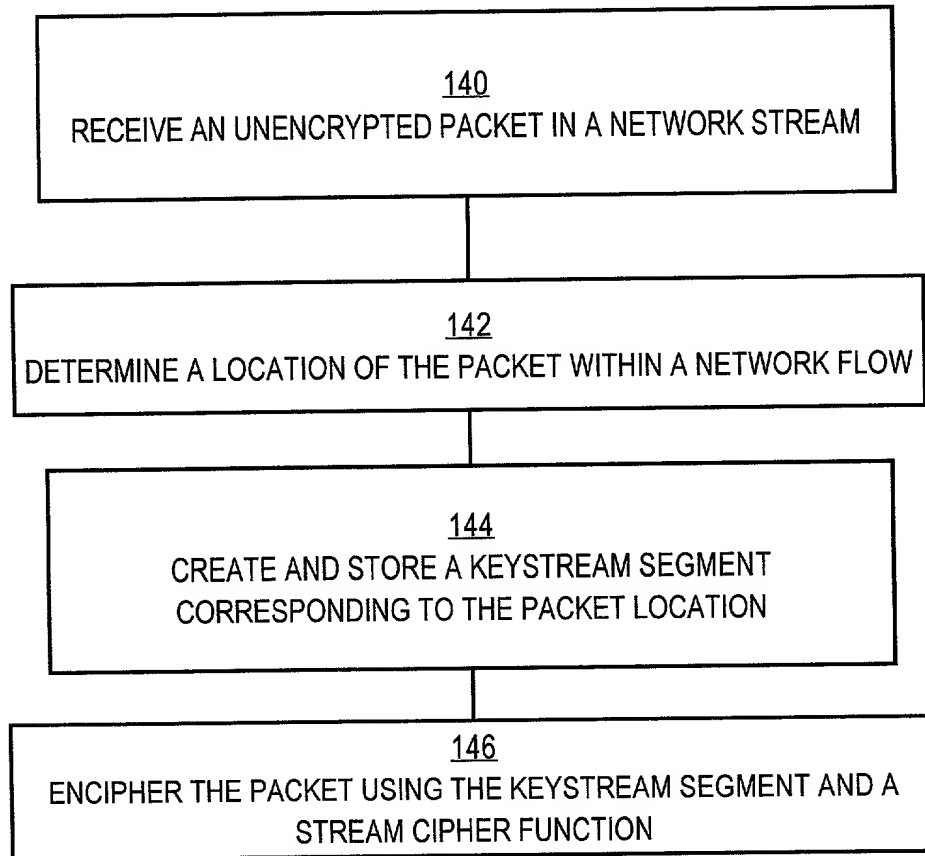


+

*Fig. 1C*



*Fig. 1D*



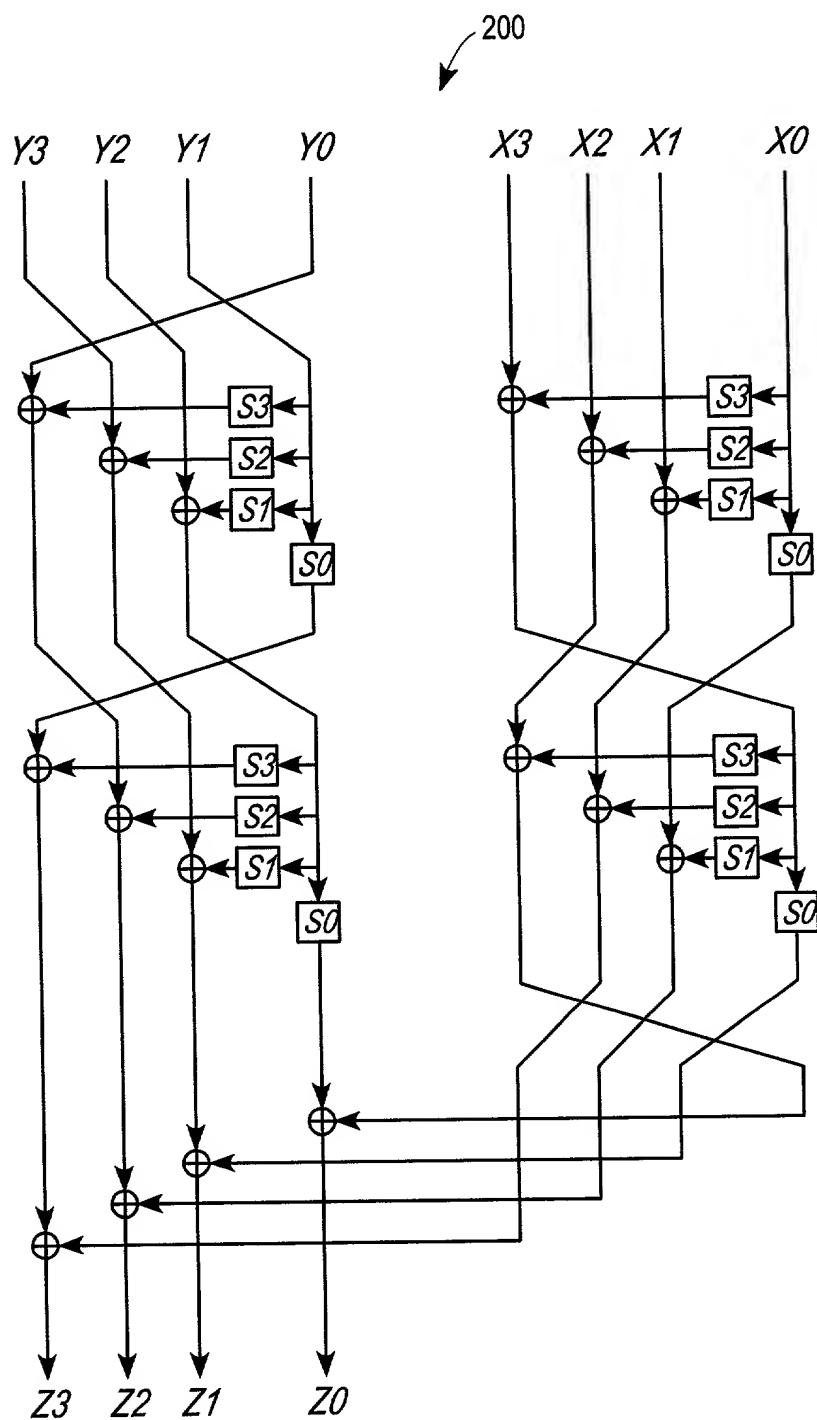


Fig. 2

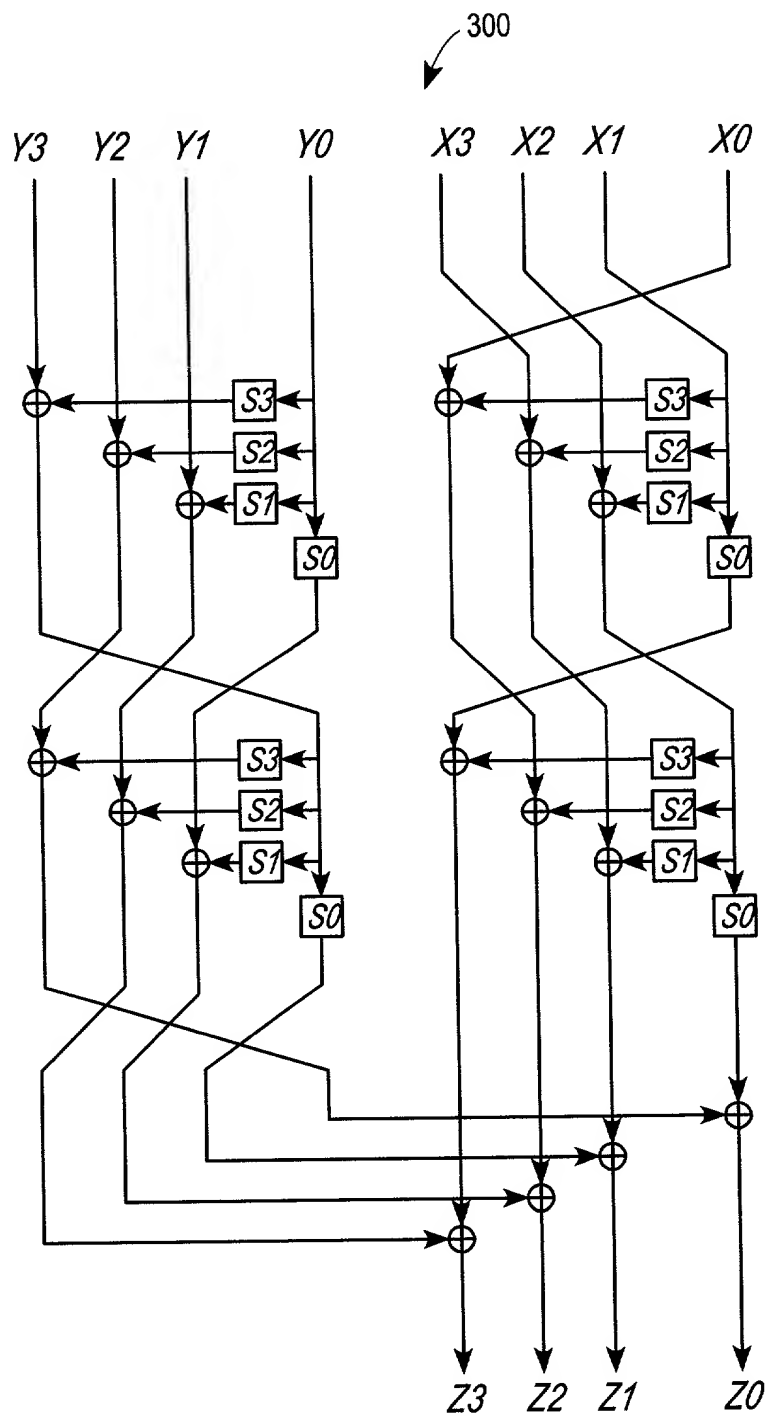
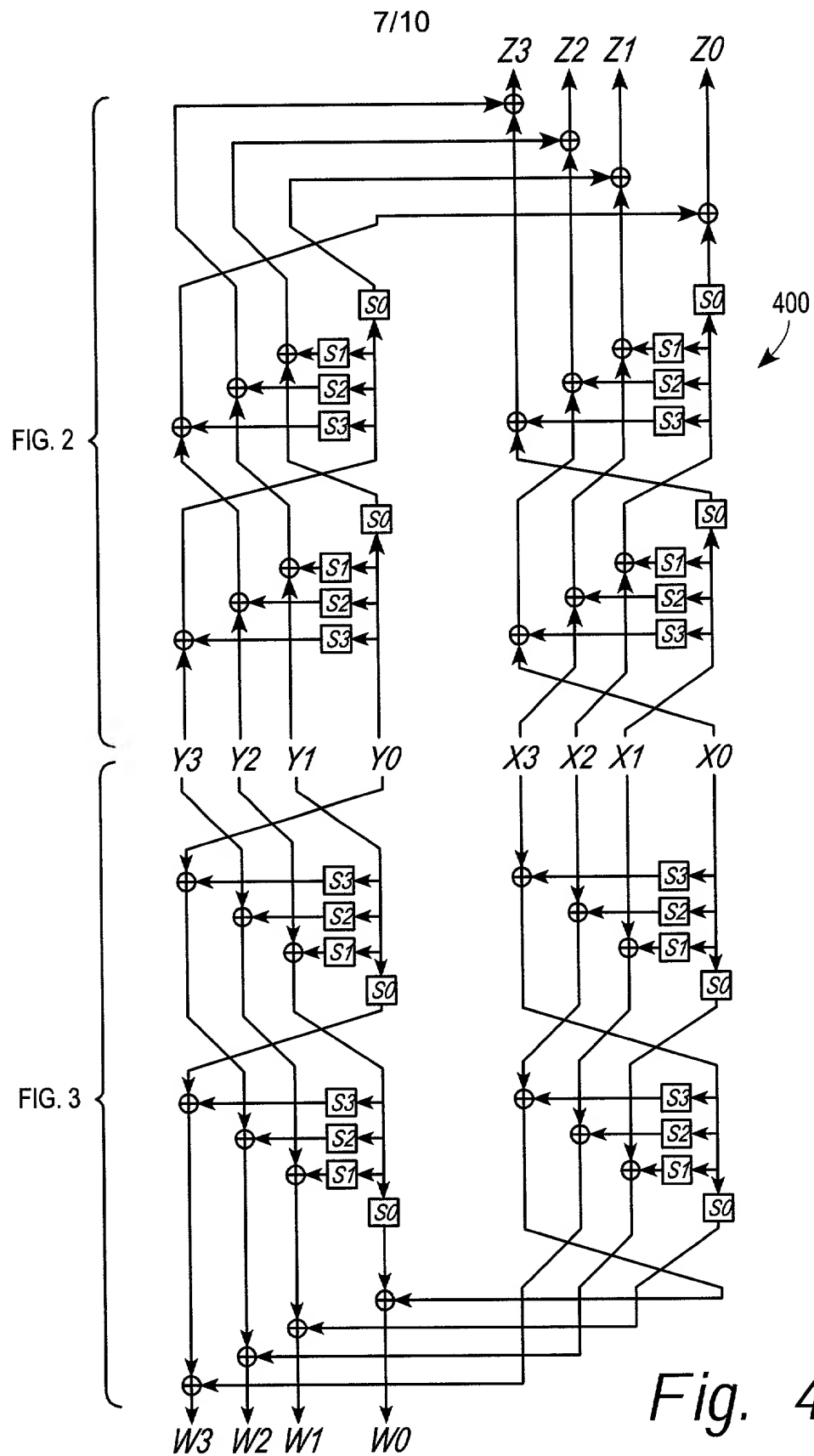


Fig. 3



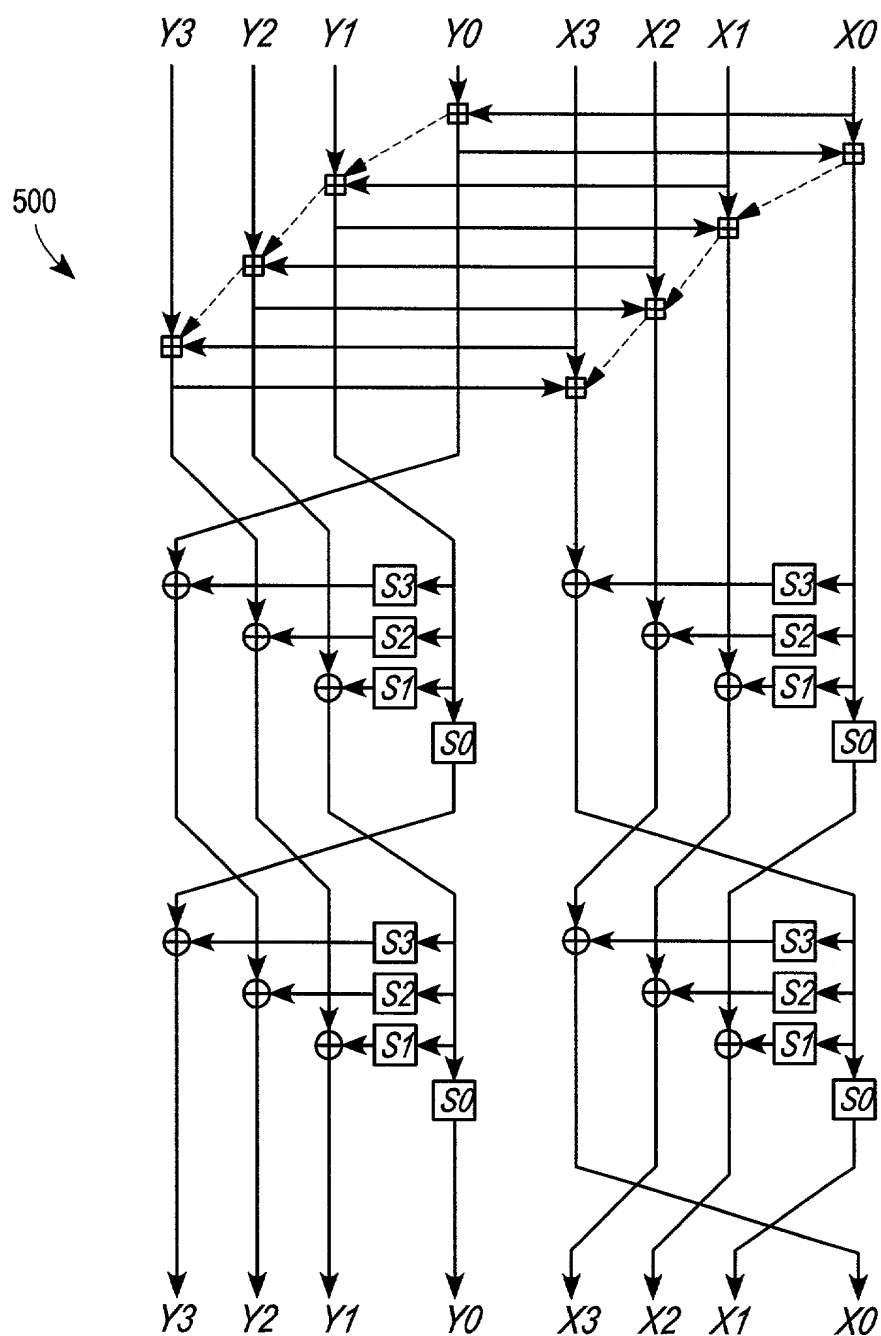
+



+

+

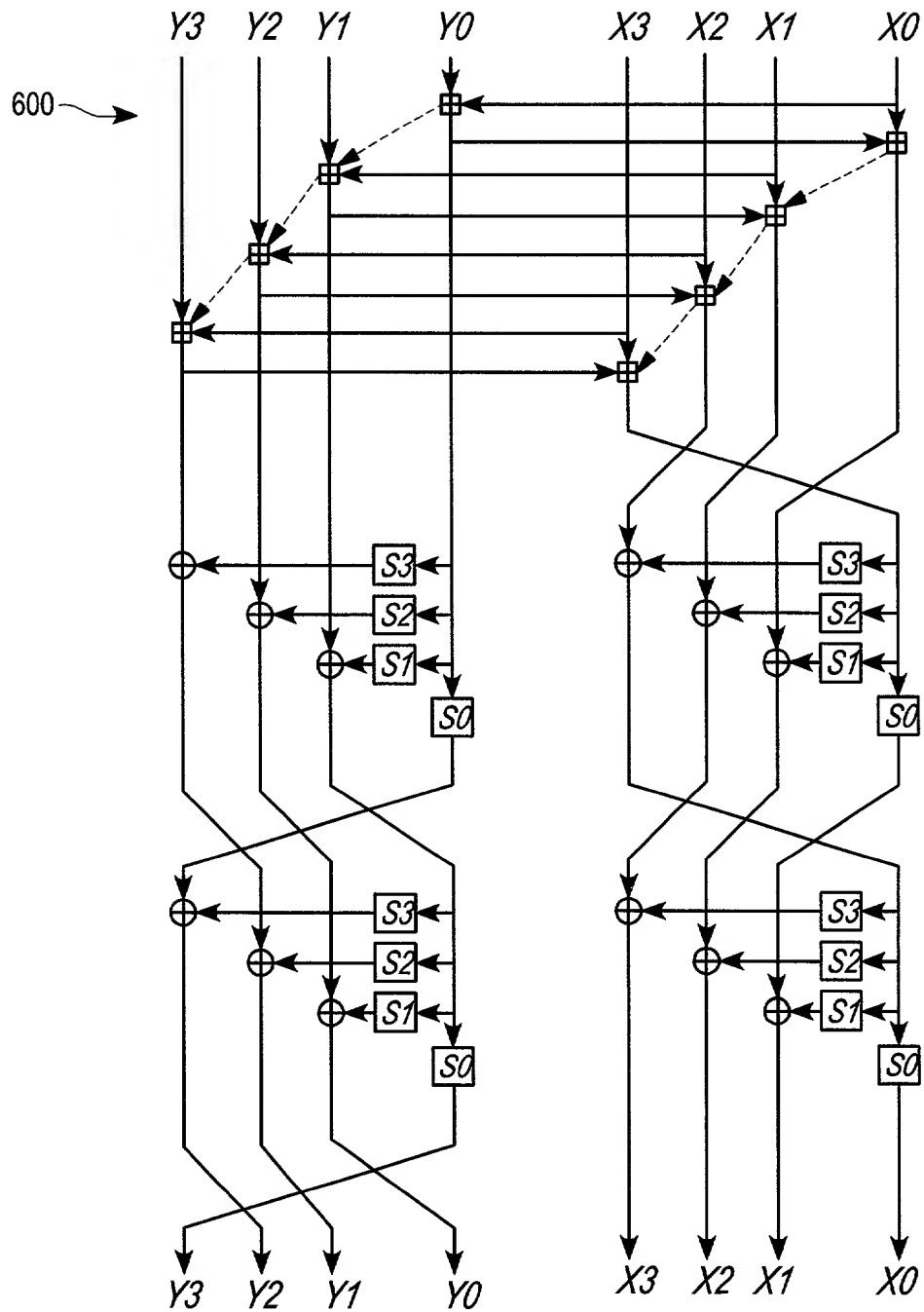
8/10

*Fig. 5*

+

+

9/10

*Fig. 6*

+

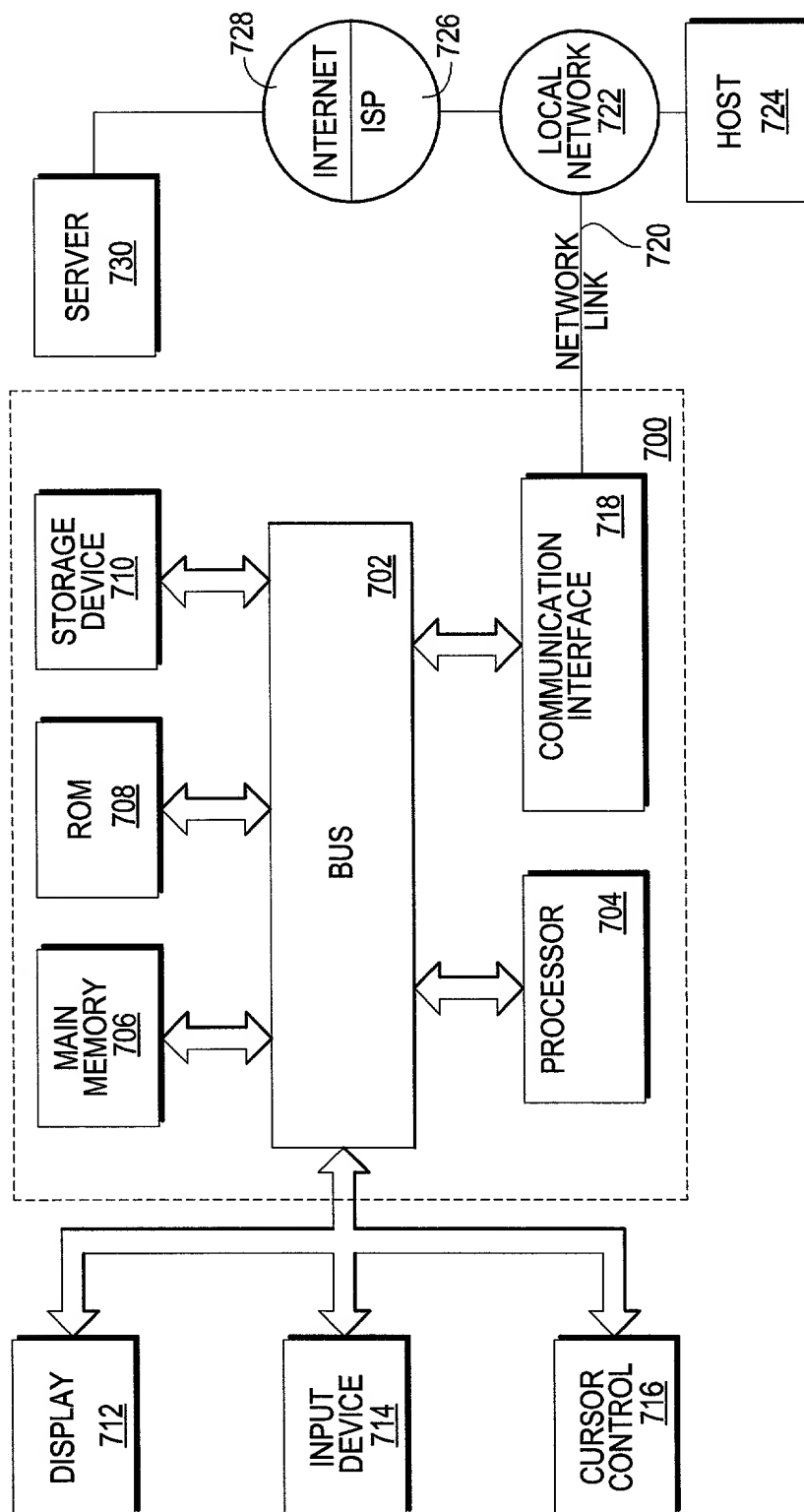


Fig. 7

Docket No.: 50325-0512

**DECLARATION AND POWER OF ATTORNEY**

As a below named inventor, I hereby declare that:

My residence, post office and citizenship are as stated below next to my name,

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter claimed and for which a patent is sought on an invention entitled **STREAM CIPHER ENCRYPTION METHOD AND APPARATUS THAT CAN EFFICIENTLY SEEK TO ARBITRARY LOCATIONS IN A KEYSTREAM**, the specification of which

☒ is attached hereto.

☐ was filed on \_\_\_\_\_ as Application Serial No. \_\_\_\_\_ and was amended on (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information that is known to me to be material to patentability in accordance with Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

**Prior Foreign Applications(s):**

Number	Country	Day/Month/Year filed	Priority Claimed
			<input type="checkbox"/>
			<input type="checkbox"/>

I hereby claim the benefit under 35 USC §119(e) of any United States provisional application(s) listed below.

**Prior Provisional Application(s):**

I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose material information as defined in Title 37, Code of Federal Regulations, Section 1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

**Prior U.S. Application(s):**

Serial No.	Filing Date	Status: Patented, Pending, Abandoned
------------	-------------	--------------------------------------


I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

I hereby appoint the following attorney(s) and/or agent(s): Edward A. Becker, Reg. No. 37,777; Marcel K. Bingham, Reg. No. 42,327; Carl L. Brandt, Reg. No. 44,555; Brian D. Hickman, Reg. No. 35,894; Christopher J. Palermo, Reg. No. 42,056; Carina M. Tan, Reg. No. 45,769; and Bobby K. Truong, Reg. No. 37,499, all of

HICKMAN PALERMO TRUONG & BECKER LLP  
1600 Willow Street  
San Jose, CA 95125-5106

with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith, and all future correspondence should be addressed to them.

Full name of sole or first inventor: DAVID A. MCGREW

Inventor's signature: 

Date: SEPT. 28, 2000

Residence: LOS GATOS, CA USA 95032

Citizenship: UNITED STATES

Post Office Address: 170 West Tasman Drive, San Jose, CA 95134

Full name of second inventor: SCOTT R. FLUHRER

Inventor's signature: 

Date: Sept 28, 2000

Residence: Sunnyvale, CA USA 94086

Citizenship: UNITED STATES

Post Office Address: 170 West Tasman Drive, San Jose, CA 95134